REMARKS

Claims 1-42 remain pending in the application. Reconsideration is respectfully requested in light of the following remarks.

Removal of Finality of Current Action:

Firstly, Applicants respectfully request removal of the finality of the present action. The § 112, second paragraph rejection of claims 1-42 is a new ground of rejection. As explained in the M.P.E.P at 706.07(a), "second or any subsequent actions on the merits shall be final, except where the examiner introduces a new ground of rejection that is neither necessitated by applicant's amendment of the claims nor based on information submitted in an information disclosure statement". The § 112, second paragraph rejection of claims 1-42 is a new ground of rejection not necessitated by any amendment by Applicants nor is it based on information submitted in an information disclosure statement. Therefore the finality of the present action is improper and must be withdrawn.

Section 112, Second Paragraph, Rejection:

The Examiner rejected claims 1-42 under 35 U.S.C. § 112, second paragraph, as indefinite. Applicants respectfully traverse this rejection. The Examiner argues that claims 1-42 are definite because "[t]here is no definition of the primary state in [Applicants'] specification" and "it was unclear what applicant means synchronize the primary state with client state." Applicants respectfully disagree and submit that claims 1-42 are not indefinite and particularly point out and distinctly claim the subject matter which applicants regard as their invention.

Contrary to the Examiner's contention, the term "primary state" is clear and easily understood by anyone of ordinary skill in the art. For example, the specification describes one embodiment of the present invention in which a primary state "may include

an instance of the session data that is globally accessible by the application servers." The specification also describes that a primary state "may be distributed on the network across one or more computer-accessible mediums." See, Specification paragraph [0028]. Additionally, the plain language of the claims (e.g. claim 1) recite a "primary state of session data configured to access by a plurality of application servers, wherein the primary state of the session data comprises a plurality of attributes." Thus, not only does the specification describe embodiments of a primary state of session data, the plain language of the claims also include specific limitations regarding the primary state of session data recited in the claims.

The plain meaning of the language of claims 1-42 is easily ascertainable by anyone of ordinary skill in the art. Applicants respectfully request removal of the § 112, second paragraph, rejection of claim 1-42.

Section 102(e) Rejection:

The Examiner rejected claims 1-42 under 35 U.S.C. § 102(e) as being anticipated by Aridor et al. (U.S. Patent 6,618,737) (hereinafter "Aridor"). Applicants respectfully traverse this rejection for at least the reasons below.

First of all Applicants note that the cited art, Aridor, has very little relevance to Applicants' invention as claimed. Aridor teaches a method for caching of read-only or non-mutable data in a clustered implementation of a Java Virtual Machine. In contrast, Applicants' claims pertain to the synchronization of application server session data in a distributed system. As will be discussed in more detailed below regarding the rejections of individual claims, Aridor's teachings do not disclose the subject matter of Applicants' claims. In fact, Aridor actually teaches away from specific limitations of Applicants' claims.

Regarding claim 1, contrary to the Examiner's assertion, Aridor fails to disclose a distributed store comprising a primary state of session data configured for

access by a plurality of application servers. Aridor fails to mention anything regarding session data or a plurality of application servers. The meanings of the terms "session data" and "application servers" are well known in the art. The teachings of Aridor have nothing to do with either session data or application servers. The Examiner refers to Aridor's master node and cites column 9, lines 25-30. However, the cited passage does not refer to session data or a plurality of application servers. In contrast, the cited passage includes definitions of terms, such as "master node, "master object" and "proxy object". In Aridor's system, the objects of a Java application are distributed among a plurality processing nodes according to a clustered Java Virtual Machine implementation. As taught by Aridor, a master node contains the master version of an object, where the master version of the object is the only version of the object on which modifications may occur. However, Aridor does not describe anything regarding a distributed store including a primary state of session data configured for access by a plurality of application servers. Aridor does not mention session data or application servers at all.

The Examiner, in the Response to Arguments, cites Aridor, column 9, line 50 column 15, lines 53-60, column 18, lines 43-53, and column 3, lines 22-64 and refers to Aridor's teachings regarding Java server applications and a configuration file supplied by a programmer. However, Aridor teaches Java applications, including server applications, which are quite different from *application servers*, as is well understood in the art. Additionally, the Aridor's configuration file is a file used by a programmer to suggest fields to be considered for caching. A file that lists fields that might be cacheable is completely irrelevant to, and fails to disclose about, **a distributed store comprising a primary state of session data configured for access by a plurality of application servers.**

Additionally, the Examiner refers to the fact that Aridor's applications "were stored in server or master node of the cluster or WAN". However, Aridor does not teach that his applications are, or are part of, an application server, as application servers are understood in the art. Instead, Aridor teaches that a cluster virtual machine for Java may

execute a Java application on a cluster in which various objects of the application are transparently distributed across the nodes of the cluster.

However, none the Examiner remarks in the Response to Arguments describes a distributed store including a primary state *of session data* configured for *access by a plurality of application servers.* Application servers and session data are particular structures well understood in the art. The mere fact that Aridor caches fields of application objects executing on different nodes does not disclose a primary state of session data configured for access by a plurality of application servers. Caches are quite different from a primary state of session data. Applicants' are not arguing that session data may not be cached. Instead, Applicants' are arguing that caching does not automatically disclose a distributed store including a primary state *of session data* configured for *access by a plurality of application servers*, as the Examiner appears to be arguing.

Thus, as noted above, Aridor fails to disclose a distributed store comprising a primary state of session data configured for access by a plurality of application servers.

In further regard to claim 1, Aridor fails to disclose a system configured to compare a client state of the session data to a benchmark of the client state to determine a subset of the attributes that have been modified in the client state. The Examiner cites column 10, lines 39-55 and refers to Aridor's micro-benchmarks. However, Aridor's micro-benchmarks are not benchmarks of a client state of session data that is compared to the client state of session to determine a subset of attributes that have been modified. Instead, Aridor uses the term benchmark in a completely different manner. Aridor describes the use of small, specialized testing and measurement programs used to gather statistics regarding the various implementations of Aridor's caching methods. Aridor teaches that benchmarks include "a tight loop in which they perform the relevant Java operation" and that "[t]he total amount of time to execute the loop is measured and divided by the number of iterations to get the amortized cost per operation" (Aridor, column 10, lines 41-42). Thus, Aridor is using a benchmark testing

program to gather information and statistics regarding his system. For example, Aridor presents TABLE 1 (top of column 11) that lists the respective amortized cost for various Java operations as measured by Aridor's micro-benchmark program. The benchmark recited in Applicants' claim 1 is of a client state of session data that is compared to the client state of session to determine a subset of attributes that have been modified, not a testing program like the benchmark in Aridor. Clearly, Aridor does not disclose comparing a client state to a benchmark of the client state to determine a subset of the attributes that have been modified in the client state.

In the Response to Arguments, the Examiner cites column 19, line 18, column 25, line 35 – column 26, line 28, column 26, line 60 – column 27, line 7, and column 29, lines 16 – 31 of Aridor. The Examiner states that Aridor teaches, "using a synchronization mechanism to modify a cached field or state on all nodes or client machine." However, as argued above, using a synchronization mechanism to modify a cached field is not the same as, nor does is disclose, comparing a client state of session data *to a benchmark* of the client state *to determine a subset of the attributes that have been modified.* The Examiner also refers to fact that Aridor teaches a mechanism to decide whether a particular field is modified to often to make caching that field beneficial. However, the mere fact that Aridor's system includes a subset of fields that are not cached because they are updated too frequently has absolutely nothing to do with the specific limitation of comparing a client state of session data *to a benchmark* of the client state *to determine a subset of the attributes that have been modified.*

The Examiner also refers again to Aridor's use of benchmark programs to measure the performance (e.g. throughput) of an application. As noted above, however, Aridor is using a benchmark testing program to gather information and statistics regarding his system. For example, Aridor presents TABLE 1 (top of column 11) that lists the respective amortized cost for various Java operations as measured by Aridor's micro-benchmark program. The benchmark recited in Applicants' claim is of a client state of session data that is compared to the client state of session to determine a subset of attributes that have been modified, not a testing program like the benchmark in Aridor.

The fact Aridor's performance testing programs are also called benchmarks does not change the fact that Aridor's benchmark programs have nothing to do with comparing a client state of session data *to a benchmark* of the client state *to determine a subset of the attributes that have been modified.*

Moreover, Aridor's system does not involve any sort of comparison between the master object, which the Examiner presumably equates the primary state of Applicants' claim, and a cached object, which the Examiner presumably equates to the client state of Applicants' claim. Instead, Aridor specifically teaches that only read-only data is cached. In some embodiments, Aridor teaches that data that may or may be read-only may be cached until the data is modified, as which time all caches of the data are invalidated. Aridor repeated stresses that modifications to data only occur at the master object on the master node. In fact, Aridor provides lengthy explanations regarding ensuring that the invalidation of all caches for a particular field occurs *prior* to updating the value of field in the master object on the master node. In Aridor's preferred embodiment, once a field is modified, the field is no longer cached and all accesses and updates are communicated to the master node for execution. In an alternative embodiment, after invalidation, the new value of a field may again be cached, but any subsequent modification again invalidates all caches of the field. Please see the following sections of Aridor: column 10, lines 1-2 and lines 15-23; column 11, lines 18 – 27 and lines 32-35; column 12, lines 59-65; column 15, lines 18-25 and lines 53-60; column 17, lines 24-32; column 18, lines 17-30; column 19, lines 33-39; column 24, lines 52-62; and column 25, lines 5-10.

In the Response to Arguments, the Examiner argues that Aridor teaches the use of a synchronization mechanism to modify a cached field or state on all the nodes or client machines. Specifically the Examiner argues, "[i]t was clearly (sic) that the synchronized process included two set[s] of data which could be server to server, server to client or client to client ... [t]he synchronized primary state or a synchronized information between server and client is not patentable."

The passages cited by the Examiner do not support the Examiner's argument. As column 19, line 18, Aridor is describing whether there is an "overriding pragmatic reason" for a candidate method should be executed on a specific node in the cluster. Aridor gives the example that "in the Java environment, it is determined whether the candidate method is a native and or synchronized method." Thus, Aridor is referring to determining whether a particular method of a Java application is a "synchronized method." A synchronized method in Java is a particular type of method that is used in synchronizing the execution of threads. Aridor is not referring to a method for synchronizing two data sets, as the Examiner erroneously contends. Similarly, the Examiner other cited passage, column 25, line 35 – column 26, line 28 does not support the Examiner's contention. At the cited passage Aridor is describing a method for re-caching data has was previously invalidated that includes an "invalidate-first observation". Aridor states that the structure of his protocol "is restricted by the independence of consistency and synchronization mechanisms in the cluster virtual machine for Java." The synchronization mechanism that Aridor discusses is not a method for synchronizing two sets of data as the Examiner contends. Instead, Aridor is discussing the fact that the modifying a data value involves first invalidating the old data on all nodes and then the new value may be cached. Aridor teaches that Java's synchronization mechanisms (e.g. mechanism to coordinate simultaneous execution of multiple threads) must be taken into account when re-caching previously invalidated data.

For instance, Aridor states:

> The invalidate-first observation implies that in a run-time system such as the cluster virtual machine for Java, where the synchronization mechanism and the consistency protocol are independently implemented, modifying a cached field at any node may be done only after invalidating the old copies on all nodes. The reason the cluster virtual machine for Java breaks the link made by the Java virtual machine between synchronization and consistency is to obtain better efficiency. Many synchronization operations (e.g. **obtaining a lock on an object used on only one node**) that would otherwise require global coordination can now be performed locally. (emphasis added, Aridor, column 26, lines 16-28).

Thus, Aridor is not referring to synchronization of two data sets, as the Examiner contends.

Aridor's system simply does not involve modifying the value of any cached field on nodes other than the master node and ensures that *all* cached versions of a modified field are invalidated prior to modifying the field on the master node. Thus, Aridor's system specifically does not involve comparing a client state to a benchmark of the client state to determine a subset of attributes that have been modified in the client state, since Aridor's system specifically guarantees that no attributes can be modified on cached versions of data. **Thus, not only does Aridor fail to disclose, Aridor actually teaches away from comparing the client state to a benchmark of the client state to determine a subset of the attributes that have been modified in the client state.**

Further in regard to claim 1, Aridor fails to disclose a system configured to synchronize the primary state with the client state according to the subset of the attributes. As described above, Aridor's system specifically prevents the modification of cached versions of data from being updated by invalidated all cached version of a modified field and only updates the field in the master object on the master node. All subsequent accesses to the modified field are remote access implemented by communicated with the master node. Thus, Aridor's system does not, by design and desire, include synchronizing a primary state with a client state according to a subset of attributes that have been modified in the client state.

Applicants remind the Examiner that anticipation requires the presence in a single prior art reference disclosure of each and every limitation of the claimed invention, arranged as in the claim. M.P.E.P 2131; *Lindemann Maschinenfabrik GmbH v. American Hoist & Derrick Co.*, 221 USPQ 481, 485 (Fed. Cir. 1984). The **identical** invention must be shown in as complete detail as is contained in the claims. *Richardson v. Suzuki Motor Co.*, 9 USPQ2d 1913, 1920 (Fed. Cir. 1989). As discussed above, Aridor clearly fails to disclose numerous specific features recited in claim 1. In fact, as shown above, Aridor teaches away from specific limitations of claim 1. Therefore, Aridor cannot be said to anticipate claim 1.

For at least the reasons above, the rejection of claim 1 is not supported by the cited art and removal thereof is respectfully requested. Similar remarks also apply to claims 10, 19, 27 and 35.

Regarding claim 3, Aridor fails to disclose a system configured to perform binary differencing of a binary representation of the client state and a binary representation of the benchmark of the client state to locate the modified attributes. The Examiner cites column 12, lines 14-21 and refers to "4-byte words". However, the cited passage is not describing anything about comparing a client state to a benchmark of the client state or about performing binary differencing. In contrast, the cited passage describes the fact that the "Java memory model guarantees atomicity for modifications at the granularity of 4-byte words." Thus, the cited passage pertains to at what granularity the Java memory model ensures that operations are performed atomically (i.e. without the possible of interruption by another operation, thread, or process). The cited passage has absolutely no relevance to comparing a client state to a benchmark of the client state or to performing binary differencing of binary representations.

In the Response to Arguments, the Examiner refers to Aridor's benchmark program "finding the shortest or different route among [a] set of cities", citing Aridor, column 30, lines 45-51. The Examiner states, [i]t was clear that a benchmark program compares the different of two set of binary data." However, Aridor's description of the Benchmark TSP program as a "parallel, branch-and-bound implementation of the well-known traveling salesman program" has absolutely no relevance to performing binary differencing of a binary representation of the client state and a binary representation of the benchmark of the client state to locate the modified attributes, as recited in Applicants' claim. Aridor is discussing the performance of his read-only caching system by using performance-measuring applications, such as the modified, parallel version of the traveling salesman program. For instance, Aridor teaches, "[u]nlike other benchmarks, read-only in practice caching (e.g. Aridor's invention) has very little impact on the benchmark TSP ... [h]owever, because there are few accesses it comes as [a]

surprise to fail to see significant benefit from read-only in practice caching in the benchmark TSP" (parenthesis added, Aridor, column 31, lines 12-16).

Neither Aridor's benchmark TSP program nor the well-known traveling salesman program has anything to do with binary differencing of binary representations of a client state and a benchmark of the client state to locate modified attributes. In fact, neither Aridor's benchmark TSP program nor the well-known traveling salesman program has anything to do comparing any type of data *to locate modified attributes*.

Furthermore, as described above regarding claim 1, Aridor's system is specifically designed to avoid modifying cached data fields, which the Examiner equates to the client state of Applicants' claim. Thus, for at least the reasons above, the rejection of claim 3 is not supported by the cited art and removal thereof is respectfully requested. Similar remarks also apply to claims 8, 12, 17, 20, 25, 28, 33, 36 and 39.

Regarding claim 4, Aridor fails to disclose a system configured to perform object graph differencing of an object graph representation of the client state and an object graph representation of the benchmark of the client state. The Examiner cites a particularly irrelevant passage (column 4, line 1) where Aridor describes smart proxies for objects. At the cited passage, Aridor teaches that given two array objects with different run-time behavior, a caching proxy may boost performance for a final static array (e.g. an array for which all access are read-only) and that remote access (i.e. no caching) should be used for an array that involves both read and write operations. Aridor does not teach or even mention anything regarding performing object graph differencing at the cited passage or anywhere else.

In the Response to Arguments, the Examiner again refers to Aridor's benchmark TSP program, citing column 30, lines 45-51. However, as described above regarding claim 2, Aridor is discussing the performance of his read-only caching system by using performance-measuring applications, such as the modified, parallel version of the traveling salesman program. Aridor is not discussing anything related to performing

object graph differencing of an object graph representation of the client state and an object graph representation of the benchmark of the client state, as recited by Applicants' claim.

The rejection of claim 4 is clearly not supported by the cited art and removal thereof is respectfully requested. Similar remarks also apply to claims 9, 13, 18, 21, 26, 29, 34, 37 and 42.

Regarding claim 5, Aridor fails to disclose a system configured to compare the tracked accessed attributes to a benchmark of the attributes of the client state to determine a subset of the tracked accessed attributes that have been modified in the client state and synchronize the primary state with the client state according to the subset of the tracked access attributes. The Examiner cites column 25, lines 48-63. However, the cited passage has no relevance to, and makes no mention of, comparing tracked accessed attributes to a benchmark of the attributes of the client state. Furthermore, as described above regarding the rejection of claim 1, this cited passage also fails to disclose anything regarding synchronizing a master version of a data field, which the Examiner equates to the primary state of Applicants' claim, with a cached version of a data field, which the Examiner equates to the client state of Applicants' claim. Please see the discussion of claim 1 above for a more detailed discussion regarding the fact that not only does Aridor fail to disclose synchronization, Aridor actual teaches away from synchronizing.

In the Response to Arguments, the Examiner contends that Aridor teaches comparing data, "including tracked attributes or tracked information" and the synchronization of modified data, citing column 26, line 66, column 13, lines 54-65 and column 26, line 20 of Aridor. However, the Examiner's cited passages have nothing to do with comparing tracked accessed attributes to a benchmark of the attributes to determine a subset of the tracked accessed attributes that have been modified in the client state and synchronize the primary state with the client state according to the subset of the tracked access attributes.

Instead, as shown below, the cited passages refer to invalidating cached objects on nodes in Aridor's clustered environment. Column 26, line 66 states that "the benefit of caching is determined by the ratio of the number of reads that in the absence of cached data would become remote compared to the number of invalidation protocols that in the absence of cached data would become simple remote writes. Column 13, lines 54 – 65 describe the whether is it better to track which nodes hold replicas of a static field or simply send the invalidate message to all nodes in light of the "obvious tradeoff between the number of invalidation messages and the amount of space required to track replicas which is a function of the cluster size." Column 26, line 20 states that the "invalidate-first observation implies that ... modifying a cached field at any node may be done only after invalidating the old copies on all nodes."

Thus, the Examiner's cited passages not only fail to support the Examiner's contention, they also fail to disclose the specific limitations of Applicants' claim 5.

For at least the reasons above, the rejection of claim 5 is not supported by the cited art and removal thereof is respectfully requested. Similar remarks also apply to claims 14, 22, 30, and 38.

Applicants also assert that the rejection of numerous other ones of the dependent claims is further unsupported by the cited art. However, since the rejection has been shown to be unsupported for the independent claims, a further discussion of the dependent claims is not necessary at this time.

CONCLUSION

Applicants submit the application is in condition for allowance, and prompt notice to that effect is respectfully requested.

If any fees are due, the Commissioner is authorized to charge said fees to Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C. Deposit Account No. 501505/5681-12000/RCK.

Also enclosed herewith are the following items:

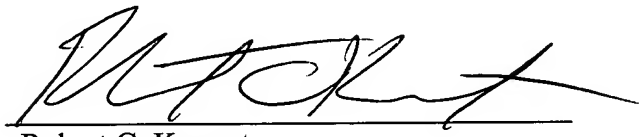
☒ Return Receipt Postcard

☐ Petition for Extension of Time

☐ Notice of Change of Address

☐ Other:

Respectfully submitted,



Robert C. Kowert
Reg. No. 39,255
ATTORNEY FOR APPLICANT(S)

Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C.
P.O. Box 398
Austin, TX 78767-0398
Phone: (512) 853-8850

Date: July 25, 2006